Chaum's Visual Voting Scheme without RPC

Marek Klonowski, Mirosław Kutyłowski, Anna Lauks, and Filip Zagórski

Institute of Mathematics, Wrocł aw University of Technology,
Marek.Klonowski@im.pwr.wroc.pl,
Miroslaw.Kutylowski@pwr.wroc.pl,
Anna.Lauks@im.pwr.wroc.pl,
Filip.Zagorski@im.pwr.wroc.pl **

Abstract. Chaum's Visual Voting is a scheme in which a voter obtains a paper receipt from a voting machine. This receipt can be used to verify that his vote was counted in the final tally, but cannot be used for vote selling - i.e., the voter cannot prove that he voted for a certain party.
The Chaum's system requires special purpose printers and application of randomized partial checking (RPC) method. RPC provides provable anonymity, but requires quite many tallying authorities.
In this paper we propose a complete design of a voting system that preserves advantages of the Chaum's scheme, but eliminates the use of special printers and RPC. It seems that in our scheme it is easier to convince an average voter that a voting machine has prepared correctly his ballot and that his vote has been counted. We also show how to achieve scalability of the system.

Keywords: electronic voting, Chaum's visual voting, re–encryption, mix

1 Introduction

There is a growing interest of electronic voting systems due to high costs, unreliability of counting results and potential frauds during traditional voting procedures. For such systems, counting the votes and collecting the results becomes easy, efficient, reliable and require less personal costs. However, there are many questions regarding the goals to be achieved - for an interesting discussion see [15]. Many nontrivial technical problems have to be solved:

1. it must be guaranteed that the technology applied does not open the doors for manipulating the votes, changing the results, ...
2. the system must be designed so that it is possible to convince a voter distrusting voting machines that his vote has been counted,
3. the receipts obtained by the voters may not be used for selling the votes.

These goals are to some extent contradictory. Resilience to vote manipulations may occur at the price of anonymity of voters. Receipts obtained from the voting machines together with information published to exclude vote manipulations may betray the choice of a voter – and so enable selling a vote. In turn, provisions against vote selling may make it hard to verify voting results.

** contact author

Chaum's Visual Voting Scheme Recently, Chaum [2] presented an idea of visual voting. The idea is a mixture of visual cryptography and processing through a cascade of mixes in order to ensure anonymity of voters. In his system, a voter has a strong evidence that each vote is really counted, even if he distrusts the infrastructure devoted to voting. Also, a voting machine cannot cheat the voter (by showing a picture that differs from the encoded vote). The voter gets a (hard-copy) receipt designed in such a way that it is meaningless for everyone, except the voter.

Understanding that the Chaum's system is flawless and provides both security and anonymity requires high mathematical skills. Certainly, an average voter cannot understand fine details of the scheme. This might cause problems with social acceptance of the scheme.

One of design goals of the system was using low cost standard hardware, but this is exactly a weak point of the scheme. The main problem is not the computing devices (standard PC's with with special open source programs borrowed from existing infrastructure of public institutions), but special purpose (and probably expensive) printers, which can print on transparent plastic cards on both sides simultaneously.

Van de Graaf proposed to abandon the idea of visual cryptography [18]. Instead, a less elegant but much cheaper solution working with regular paper printers has been proposed. In this solution, one can use the existing infrastructure without significant additional expenses. The ballots in the system of Van de Graaf are designed differently, votes are prepared in advance by special authorities. Thanks to ballot's modification, expensive printers are no longer needed, but other problems arise. The scheme requires existence of two authorities that will not cheat together. The ballots have to be distributed among the voting machines. One of the main features of the Chaum's scheme is that the votes are prepared alone by the voting machines does not hold anymore.

As a subroutine for ensuring correct vote decoding, Chaum's system uses *Randomized Partial Checking* (RPC) technique introduced by Jakobsson et al. in [11]. RPC applied to a cascade of mixes reveals a large number of connections between the input and the output of each mix. Nevertheless, it was proved [7] that very high anonymity level is achieved for any number of votes processed with $O(1)$ mix-servers only. Even if the number of mix-servers is $O(1)$, the problem is that it might be too high for practical implementation. The point is that each mix requires some public-key ciphertext to be printed on a ballot, and it may turn out that in order to achieve some anonymity level we need to print more data that actually can be placed on the receipt.

New results We design a secure and fairly practical system of voting based on electronic voting machines, in which we combine ideas of Chaum's visual voting, printing method of Van de Graaf and onion-like encryption technique presented in [12].

Our solution has the following important features:

low cost: the whole infrastructure requires only fairly standard devices: scanners or regular bar-code laser readers, and paper printers; thus we avoid sophisticated printers that are necessary for the Chaum's scheme,

efficiency: computational effort of performing the whole procedure is linear with respect to the number of voters.

scalability: processing of votes in the scheme presented by us can be parallelized. In that case provable anonymity of votes is preserved (such a parallelization for the Chaum's scheme based on RPC method is more problematic). This causes not only speed-up and lower load per server, but also decreases probability of undetected cheating by a mix-server. Unfortunately, parallelization has a negative impact on the level of anonymity. But, it turns out that for most cases (see section [5]) it ensures a higher level of anonymity than with Chaum's visual voting.

voter verifiable elections: every voter can verify that with an overwhelming probability his vote is in the final tally,

vote selling: nobody can sell votes without cooperation with a voting machine or all tallying authorities,

cheating detection: every tallying authority that cheats during vote decoding is caught with an overwhelming probability,

low number of tallying authorities: the number of tallying authorities necessary for decoding the votes can be reduced significantly compared with the Chaum's solution based on the RPC method.

The main disadvantage of the scheme presented in this paper is that it does not work in practice, if the number of candidates is large - it becomes impossible to encode all necessary information on a single paper receipt given to the voter. This is not a problem for presidential elections in the USA, but it is a problem for parliament elections in Poland.

Voter-verifiable voting schemes One can regard a voting process as a problem of submitting messages $v(x_i)$ to a kind of bulletin board by voters $x_1, ..., x_N$ in such a way that:

- every x_i can verify if $v(x_i)$ is delivered to the bulletin board,
- it is infeasible to link x_i with his vote; even if x_i is cooperating, it is infeasible to build a convincing proof that x_i voted in a particular way.

Let us recall the idea of Chaum's [2] solution for this problem. Every x_i chooses $v(x_i)$ and gets a receipt $r(x_i)$. The receipt is constructed in such a way that after passing through λ mix-servers and re-coding at each server it becomes $v(x_i)$ again. We assume that a voting machine that prepared $r(x_i)$ from $v(x_i)$ is honest. When all messages are delivered, then it is easy to count the election result. In order to avoid election fraud by the mix-servers, every mix-server must reveal a randomly chosen half of it's operations during Randomized Partial Checking procedure. The connections to be revealed are chosen by an independent party – for instance an opponent of the authority ruling a particular mix-server. Probability for the cheating server to be caught is $\frac{1}{2}$, if a single vote is manipulated. Probability of successful cheating k votes is lower than to $\frac{1}{2^k}$

Our solution is different. Two encoded and different copies of $v(x_i)$ and two encoded receipts are sent through mix servers. The probability of successful delivery depends on the number of messages. In order to cheat, a mix-server has to guess which out of $4N$ messages are ciphertexts that are logically connected to x_i. To succeed in cheating, the mix-server has to replace each copy of $v(x_i)$, but none of $r(x_j)$. The probability

of undetected removing k votes in our system depends on the number of votes N being sent and is less than $1/(2N)^k$, compared with $1/2^k$ in the Chaum's system.

Another approach of checking correctness of mix-servers' behaviour is introduced in [16]. In this paper, a method of shuffling El-Gamal ciphertexts is presented - after processing a whole batch of ciphertexts, each of mix-servers proves that its output represents a permutation of properly re-encrypted input elements. In our approach, any fraud is detected automatically and only in that case some additional computations are required for finding dishonest servers.

Let us mention yet another solution. Recall that it is possible to generate ciphertexts so that re-encryption of ciphertexts is possible without knowledge of public/private keys [9]. Moreover, it is possible to sign such ciphertexts so that the signatures can be re-encrypted just like the ciphertexts [14]. If votes are encrypted and signed in this way by respective authorities, then mix-servers processing the encoded votes can re-code the votes and the signatures. At the same time signature of each encoded vote can be checked. The only problem is duplicating the votes, but we can apply similar techniques as presented in this paper to cope with this problem. The main difference with the schemes discussed above is that the votes must be prepared in advance by a trusted third-party (or parties), like in the van de Graaf's scheme.

2 Building Blocks

Mix-servers Mix-server is a basic anonymity primitive introduced by D. Chaum in [3]. A mix-server takes a batch of encrypted messages and outputs them after recoding in a random order. The recoding procedure can be, for instance, decryption. It must hide any link between inputs and outputs of the mix-server. The main goal of a mix-server is to hide the correspondence between the input and the output batch. Since then, many papers addressed attacks on mixes and message tracing, reputability of mixes, composing networks of mixes, ...

RE-Onions with Recoding We describe now an encoding scheme, which is a simplified version of URE-onions from [12].

We create an onion-like structure, which we call ***RE-onion***, based on ElGamal encryption in such a way that λ parties have to process the RE-onion before it is finally decrypted. An RE-onion shall be used to send a message m through a mix cascade of λ servers. For $1 \leq j \leq \lambda$, let y_j be the public key of the jth mix, and let x_j be the corresponding private key, that is, $y_j = g^{x_j}$. In this formula g is a generator of a group with hard discrete logarithm problem.

In order to prepare an onion we choose a string k_1 uniformly at random. Then an onion is computed as:

$$(\alpha, \beta) := (m \cdot (y_1 \cdot \ldots \cdot y_\lambda)^{k_1}, g^{k_1}) \ .$$

When such an onion (after some decoding and re-encryption) is delivered to mix i, then it has the form

$$(\alpha_i, \beta_i) = (m \cdot (y_i \cdot \ldots \cdot y_\lambda)^{k_i}, g^{k_i}) \ .$$

Then the onion gets partially decrypted and re-encrypted – the following operations are executed with a randomly chosen r_i:

$$\begin{aligned}
\alpha_{i+1} &:= \alpha_i / \beta_i^{x_i} \ , \\
\alpha_{i+1} &:= \alpha_{i+1} \cdot (y_{i+1} \cdot \ldots \cdot y_\lambda)^{r_i} \ , \\
\beta_{i+1} &:= \beta_i \cdot g^{r_i} \ .
\end{aligned}$$

It is easy to see that after performing these operations we get

$$(\alpha_{i+1}, \beta_{i+1}) = (m \cdot (y_{i+1} \cdot \ldots \cdot y_\lambda)^{k_{i+1}}, g^{k_{i+1}})$$

where $k_{i+1} = k_i + r_i$.

Opening an Onion We use a trick borrowed from [2]. When we construct an RE-onion, then we need a random exponent. This exponent is generated by a strong pseudo-random number generator $\mathcal{R}$ from a seed $s(q)$, where $s(q)$ denotes a signature over q. The signature scheme used is deterministic (that is, we can use RSA, but not ElGamal at this point). The string q is also (pseudo)random and is stored together with the RE-onion created.

Notice that it is impossible to recover the exponent used for constructing $(\alpha, \beta) = (m \cdot (y_1 \ldots y_\lambda)^{k_1}, g^{k_1})$ given q only. Indeed, this would require finding $s(q)$, but this is impossible without knowledge of the signing key. However, in the case that the onion creator would like to show the contents of the onion it suffices to publish $s(q)$. Then everybody can reconstruct $\mathcal{R}(s(q))$, derive the exponent k_1, and finally derive $m := \alpha/(y_1 \ldots y_\lambda)^k$ and check whether $\beta = g^k$.

3 Description of the Voting Protocol

Overview In the early morning of the election day, each voting machine generates a couple of key pairs for signature schemes used. No cryptographic material is imported from outside – such as pre-prepared ballots, except the public keys of the tallying authorities running mix-servers. A ballot is created online by the voting machine once a voter starts the voting procedure.

When the voter's identity and his right to vote is confirmed, he is is admitted to a voting machine. During the procedure the machine prints some codes in three phases. During the first phase some codes are printed on the paper - but the ballot is still not released to the voter. Then the voter makes his decision about his vote. Additionally, the voter chooses the *left* or the *right* side of the ballot for checking its correctness. At this moment some codes are overwritten, additional information is printed and the ballot is released to the voter. The information from the ballot (without overwritten stuff) is scanned at a different stand during the next step - and the voter gets a confirmation that his ballot has been inserted into the counting system.

The voter can check integrity of his ballot at any machine equipped with a scanner (it may be provided outside by any "watch dog" organization). The control procedure provides a proof that the half of the ballot chosen for control was prepared correctly.

The codes read from all ballots are gathered and are processed through a cascade of mixes run by tallying authorities. They can be published, but this is not necessary for proving correctness of the election result.

Voter's preferences are encoded in four RE-onions read from the ballot. Two RE-onions are used to encode the vote and two RE-onions are used to encode the identifier associated with this ballot. The identifiers do not reveal any information about the voter's choice. Each of these onions is processed by the mix-servers. This includes partial decryption and re-encryption. It is a key feature that the onions of both kinds cannot be distinguished until the final recoding occurs. Then we get a plaintext of the vote and an identifier of the ballot.

Finally, the lists of identifiers and votes obtained are published. Then a voter can check, if the identifier of his ballot appears on the list. If the list of identifiers has been published prior to mixing, one can compare both lists. If some identifier is missing, then it is an evidence of faulty processing of votes. In this case an investigation is started, during which a dishonest mix-server is identified. A similar procedure could be started in the case when a vote gets corrupted. However, we refrain from an investigation in this case since it could break anonymity of the voter. However, the point is that even if the mix server can identify four RE-onions coming from a single voter, it has $\frac{1}{6}$ chance to remove the onions encoding the identifier. Of course, we can send more than four RE-onions to obtain even greater probability of detection of the dishonest mix-server, but this influences the size of the ballot (and the number of codes printed) as well as the amount of work for the mix-servers. Moreover using four onions provides very high level of security.

Each mix-server is under control of a different party. There must be a provision (such as key escrow) that enables to recover the private key of a mix-server in the case when dishonest behavior of this mix is detected.

Preparation of a Ballot We describe a single voting machine V. It has two pairs of keys of a digital signature algorithm. Let K and K' be the private keys of V and K_{pub} with K'_{pub} be the public keys related to K and K'. It has also a serial number ser_V.

The ballot printed by voting machine V will contain two sides, which we call the *left side* and the *right side*. Each of sides has two columns - columns 1 and 2. This is for the sake of exposition only, the codes can be printed in a different manner).

For the sake of simplicity assume that the voter can choose between two parties - the Blue Party and the Yellow Party. The ballot can contain (depending on the voter's choice) the following components:

- a ballot identifier r, which is a random string signed by the voting machine,
- an auxiliary string q,
- RE-onions B_1^R, B_2^R, Y_1^R, Y_2^R, I_1^R, I_2^R on the right side and RE-onions B_1^L, B_2^L, Y_1^L, Y_2^L, I_1^L, I_2^L on the left side, with the corresponding labels B, Y, I,
- random strings r_L and r_R, signatures $\mathrm{sig}_K(q, 1, L)$, $\mathrm{sig}_K(q, 2, L)$, $\mathrm{sig}_K(q, 1, R)$, $\mathrm{sig}_K(q, 2, R)$ – but only one string and one signature corresponding to the side and column chosen for verification of ballot's correctness).

The string q is used for constructing the onions. Namely, while constructing the onions B_i^X, Y_i^X, I_i^X ($i \in \{1, 2\}, X \in \{L, R\}$), the signature $\mathrm{sig}_K(q, i, X)$ is created

(sig is a deterministic signature scheme and K is the private key of the voting machine used for signing the onions). Then a pseudorandom number generator $\mathcal{R}$ with the seed $\text{sig}_K(q, i, X)$ is used to generate the exponents used in the construction of these RE-onions.

For $X \in \{L, R\}$, the onions B_1^X, B_2^X encode a vote for the Blue Party, while the onions Y_1^X, Y_2^X encode a vote for the Yellow Party. The onions I_1^X, I_2^X encode the identifier r. After full decoding of the onions we get, respectively for $i = 1, 2$:

- $(B, r_X, ser_V, \text{sig}'_{K'}(B, r_X, i))$ from B_i^X,
- $(Y, r_X, ser_V, \text{sig}'_{K'}(Y, r_X, i))$ from Y_i^X,
- $(r, ser_V, \text{sig}'_{K'}(r, i, X))$ from I_i^X.

for some random r_L, r_R chosen for this ballot. The private key K' is used only for signing the votes and identifiers with signature scheme sig$'$. The onions are RE-onions encoded with the public keys of the tallying authorities.

The codes of all onions are printed on the ballot. Each pair of onions encoding the same information is clearly marked as a pair. However, the ordering of pairs is random. The orderings chosen for the left and the right side are stochastically independent.

Making Choices Once a ballot consisting of the strings listed above is created, the voter makes a couple of choices:

1. which side will be used for checking and which for making a vote,
2. the choice of a column on the checked side that will be examined,
3. the vote for particular party.

Assume that the voter has chosen for checking the left side containing B_1^L, B_2^L, Y_1^L, Y_2^L, I_1^L, I_2^L. Then, r_L is printed (see Fig. 2).

The next choice of the voter is $j \in \{1, 2\}$. After making this decision by the voter the printer performs the following steps (see Fig. 3):

- B_j^L, Y_j^L, I_j^L get overwritten. (This excludes the use of the votes encoded on the left side, since for each choice one half will be missing.)
- For $j' \in \{1, 2\}$, $j' \neq j$, the voting machine *opens* $B_{j'}^L$, $Y_{j'}^L$, $I_{j'}^L$. That is, it prints $\text{sig}_K(q, j', L)$.

In the next step, the user chooses how to vote. The voter's preference is encoded on the other side (this time the right side). The printer overwrites the onions that encode the vote for the party that has not been chosen. That is, if the Blue Party has been chosen by the voter, then Y_1^R, Y_2^R are overwritten, otherwise B_1^R, B_2^R are overwritten. Simultaneously, all labels indicating which onion correspond to which party are overwritten.

When printing is finished, the ballot is released to the voter.

Checking a Ballot The voter can check consistency of information on the side chosen for checking. He opens the onions on this side using the signature printed and checks that their contents adhere to the protocol. For instance, he checks that the encoded votes corresponds to the labels printed,that the identifier r_X is properly contained in the votes, and that the onion contents are properly signed. Checking is done by an appropriate program after reading in the codes from the ballot.

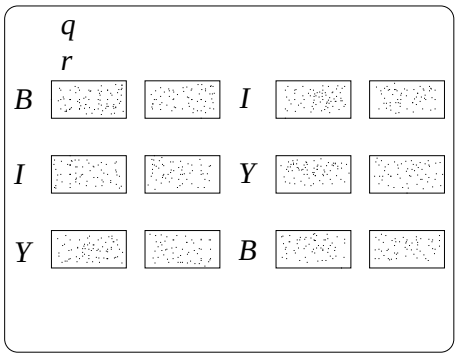

Fig. 1. Initial shape of the voting card as seen by a voter at the first step

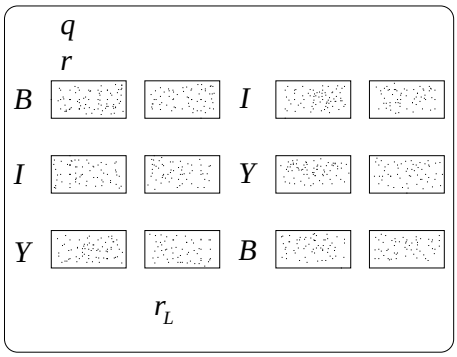

Fig. 2. The voting card after a voter chooses the left side to be verified - r_L gets printed

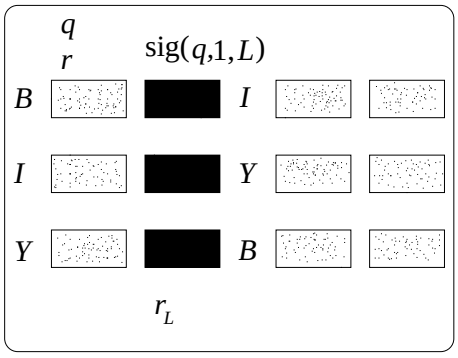

Fig. 3. The voting card after choosing the column to verification, additionally $\text{sig}_K(q, i, X)$ is printed

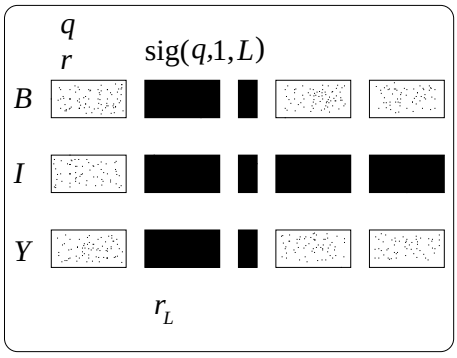

Fig. 4. The voting card after choosing a party; information betraying the voters preference and the unused onions get overprinted

Counting Procedure The voter presents his ballot to a scanner that reads two pairs of onions that are left on the side containing the vote. At the same time it checks the signature of the voting machine contained in r (if the signature is invalid, the voting card is rejected). A confirmation is printed on the ballot confirming that the ballot has been registered (it also prevents using the same ballot twice). All vote identifiers r may be published before counting of voices begins.

After reading all ballots the onions are sent to the first tallying authority. It recodes the RE-onions as described above and gives the result to the second tallying authority. The second authority recodes the onions and gives the result to the next authority. This process is continued until the last tallying authority finishes decoding. For the purpose of a future investigation (which is necessary when the final output is faulty), if one authority gives (recoded) onions to another authority, then both authorities sign the list of these onions and retain it safely.

The last tallying authority publishes the list of the strings read from the onions from the final decoding. If every participant behaves according to the protocol, then the list contains:

- pairs of signed identifiers of the form

$$(r, ser_V, \text{sig}'_{K'}(r, 1, X)), (r, ser_V, \text{sig}'_{K'}(r, 2, X))$$

- pairs of votes of the forms

$$(B, s, ser_V, \text{sig}'_{K'}(B, s, 1)), (B, s, \text{sig}'_{K'}(B, s, 2))$$
$$(Y, z, ser_V, \text{sig}'_{K'}(Y, z, 1)), (Y, z, \text{sig}'_{K'}(Y, z, 2))$$

  for random identifiers s, z.

The identifiers of the ballots (voting card) are checked either by the voters or against the list published before counting procedure starts. If both lists contain exactly the same identifiers, all signatures are valid, the number of votes and the number of identifiers are equal to the number of the voters participating in the elections, then the votes are counted and the election result is announced.

Investigation procedure Let us assume that after decoding the last tallying authority gets a string that is neither a valid vote nor a valid identifier (i.e. a signature of a voting machine is invalid or missing). In this case the route of the faulty message m should be traced back in order to find the authority responsible for a manipulation.

First, the last server presents the ElGamal ciphertext from which it has obtained m, say (a, b). Then it proves that $a/m = b^x$, where x is the private decryption key of this authority. More precisely, a proof of equality of discrete logarithms [17] for pairs $(a/m, b)$ and (y, g) is shown, where y is the public key of the authority.

If the onion pointed to by the last authority is on the list of onions it has got from the previous authority, then it is the turn of the previous authority to prove its good behavior. The procedure is the same way as in the case of the last authority, except that instead of decryption we consider partial decryption. Additionally, the authority shows which input RE-onion was re-encrypted to obtain the faulty RE-onion. For this purpose, the

authority publishes the exponent used for re-encryption. Note that it is not necessary to store all exponents used for re-encryption – they might be derives from a secret key with a strong pseudorandom generator. If this authority proves that it has properly processed the onion containing the faulty m, then the next authority must prove its source of m. This procedure is continued until we come to the point that some authority cannot prove to be not guilty.

The same investigation takes place, if the final list contains duplicates. In this case we trace back each of the onions holding the duplicate message. An authority is found guilty, if it can show only one source of onions that are decoded to the same string.

Improper behavior Let us consider different possibilities of misbehavior of the mixes:

Removing an onion: in this case the number of onions in the input and in the output of a tallying authority disagree; the fault is immediately discovered provided that the number of onions (and votes!) is recorded at each stage.

Inserting a new onion: nobody except the voting machine can prepare an onion that will be correctly decoded. Indeed - the message obtained by the last tallying authority need to be signed by the voting machine at each case. In this case the last tallying authority starts an investigation described above. It shows the authority that has injected a new message.

Duplicating an onion: Thanks to re-encryption features, a duplicate can be easily hidden. However, on the final list we get two identical strings. If the duplicated string is an identifier, then an investigation is started and one of the authorities is found guilty. If the duplicated string is a vote, then we do not start an investigation for this vote, since it could endanger voters anonymity. However, even in this case we have a chance to catch an authority that has duplicated the vote.

In order to succeed in cheating, one has to replace a pair of onions which encode votes but not identifiers. Having 4 onions, the probability of a successful replacement is equal to $\frac{1}{6}$ ($= \frac{2}{4} \cdot \frac{1}{3}$). So in that case the probability that an onion holding an identifier has been deleted is $\frac{5}{6}$. Then, with high probability, some vote identifier is missing from the final list and we may start an investigation that traces the route of an onion holding this identifier.

When the authority, which performs a fraud gets $4N$ onions, the probability of duplicating $2k$ onions encoding votes is equal to:

$$\frac{2N}{4N}\frac{1}{4N-1}\frac{2N-2}{4N-2}\frac{1}{4N-3}\cdots\frac{2N-4k}{4N-2k+1}\frac{1}{4N-2k} \leq \frac{(2N)^k}{(4N-2k)^{2k}} \leq \frac{1}{(2N)^k}$$

Manipulating an onion: Since each vote and identifier is accompanied by a signature of a voting machine, no one can derive a new onion with a valid contents, except for creating an onion with the same contents as another onion (otherwise, we would have a procedure breaking the signature scheme used). So if an onion is manipulated, then after the last decoding we get an invalid message. In this case an investigation is started to trace back the route of an onion containing this message.

Hacked voting machine: There is a possibility that a mallware is running on a voting machine (remember that we use public infrastructure). In this case the voting cards may be faulty. However, thanks to the verification procedure it will be detected by each voter with probability at least $\frac{1}{4}$.

Dishonest comission: There is also a possibility that there is dishonest group of people supervising a voting machine. They may use the machine for casting extra votes. Since the points, where the votes are get registered, are checking only the signatures of the voting machines, the manipulation get undetected at the first moment. However, finally each decoded vote reveals from which voting machine it came. If the number of votes from this machine does not agree with the list of voters that participated in elections the manipulation is revealed. Then it is easy to recompute results of voting and repeat the voting procedure only in that commission, where the problems have occurred, and not in the whole country.

One may be tempted to redesign the scheme in order to hide also on which voting machine a vote has been generated. This may cause a lot of problems related to dishonest commissions. Another point is that, at least in our country, it is required by law that the election results from each commission are published.

4 Improvements

Overprinting some regions of the voting card has one major advantage – it is quite easy to convince an average voter that the voting machine is caught with probability $\frac{1}{2}$ if it cheats. We find that very important due to the social acceptance of the election process.

There are some drawbacks of this approach. Firstly, an encoded onion requires some space on the voting card. This may be problematic when we consider multi-candidate elections. Even more serious problem is that a voter may bring a camera to the voting booth and take photos of the voting card during the voting process. Those pictures can be used for selling votes. The same problem appears also in the Chaum's scheme.

There is another issue related to taking pictures of the voting cards. If a dishonest commission at the polling place takes pictures of voting cards, then it gets onions encoding other voter's choices. Then the commission can replace the onion encoding the voter's choice with another onion of their preference. Such a cheating will not be detected, since the commission can prepare spoofed votes by using proper identifiers combined with onions chosen of its choice (retrieved from the picture). In the Chaum's scheme it is impossible to cheat in such a way. Of course, for all voting schemes, installing a camera at the voting booth always destroys anonymity of a voter.

In our opinion it is difficult to guarantee that no camera can be taken to the voting booth (even if it is forbidden by law and there is some control before entering the booth). For that reason we propose a simple modification of the system that solves problems mentioned above. On the other hand, we suppose that it is little bit less intuitive for voters without sophisticated knowledge. So it can be harder accepted by an average voter.

Modified Scheme Instead of printing all onions during the first phase, the commitments of these onions are printed (for instance hash values of these onions). After making the choice by the voter, certain onions are printed - namely these onions that are not overprinted according to the original scheme. In this way there is no need to overprint any onion. This approach not only saves the space on the voting card (a commitment is much shorter than an onion – this can be crucial due to space limitations), but it

also provides a higher level of security. Namely, recording of even the whole vote-preparation process does not allow manipulating the votes as it was described above.

For the modified scheme the verification process contains one more step, namely checking the commitments of the onions that appear on the receipt.

5 Scaling the solution

Processing all ballots through a single cascade of mix-servers might be problematic in some practical situations. Instead we could employ a parallel cascade of mixes. In this way the number of codes processed by each mix-server may be reduced, a low cost server may be used and processing speed might be increased.

There are many possible architectures for the networks of mixes, we shall discuss here a *parallel mix cascade* [10]: it consists of some number of layers, where each layer consists of k mix-servers. The output of each mix is divided into k equal parts and the jth part goes into the jth mix of the next layer.

In our system allowing messages being processed in parallel by different mix-servers might have a positive impact on the overall security of the system. The probability that a pair of onions encoding the same vote is processed by the same mix-server is equal to $\frac{1}{k}$ (with k servers in a layer), so the probability of success in replacement attack is even lower than in a sequential processing if a single mix-server on a layer is dishonest. Unfortunately, there is another issue. Processing the codes in parallel has a negative impact on anonymity. The following arguments are not strict, but allow to estimate a guaranteed level of anonymity for different architectures of networks of mixes. We consider following instances:

1. a mix cascade running our algorithm,
2. a parallel mix cascade running our algorithm,
3. a (parallel) mix cascade running with RPC as a verification procedure used.

Assume that there are N messages traveling through the system. Consider a parallel mix cascade consisting of l layers with k mix-servers on each layer. Every mix-server processes the same number of messages, namely $\frac{N}{k}$, and it passes $\frac{N}{k^2}$ messages to each mix-server of the next layer. Assuming that each mix-server produces every permutation of messages with the same probability, we ask how many layers do we need so that we can get any permutations after the last layer. In fact, we compute in how many ways a specific permutation of messages can be achieved by the parallel mix cascade.

Every server can permute the messages in $(\frac{N}{k})!$ different ways. There are k servers in a layer, so the number of possible permutations produced by a single layer equals $\left((\frac{N}{k})!\right)^k$. Hence the number of possible settings of the whole cascade is $\left((\frac{N}{k})!\right)^{kl}$. Then a value $z(N,k,l) = \left((\frac{N}{k})!\right)^{kl}/N!$ is the average number of ways in which a specific permutation of messages can be obtained by passing through the parallel mix-cascade. When we compute coefficient z in an analogous way for a mix cascade consisting of l mixes, we get $z(N,1,l) = (N!)^{l-1}$. For simplifying the following computation, we approximate $n!$ by $(\frac{n}{e})^n$. Using this formula we can estimate ,,how much anonymity we lose" while using parallel mix cascade instead of a sequential mix cascade. The ,,loss"

factor is equal to $\frac{z(N,1,l)}{z(N,k,l)} \approx k^{Nl}$. So it seems that parallelization is not good idea, but let us consider also a similar coefficient in the context of RPC. Now, every server can also output messages in $(\frac{N}{k})!$ different orderings, but after all it has to reveal half of the connections, so in fact, there is only $(\frac{N}{2k})!$ possible orderings. Comparing coefficient z for a sequential mix-cascade with RPC with parallel mix cascade we get

$$\frac{z(N,k,l)}{z_{RPC}(N,1,l)} \approx \left(\frac{2N}{ek^2}\right)^{\frac{Nl}{2}} .$$

So, for $k \leq \sqrt{2N/e}$, parallelization in our system makes less harm for anonymity than usage of RPC for a sequential mix cascade. For example, for $N = 1000$ up to $k = 27$ parallelization is better. It is reasonable to assume that for the parameters occurring during elections it is always the case that $k \leq \sqrt{2N/e}$, we see that the usage of RPC is always less attractive than parallel mix cascades.

Finally, let us remark that parallel mix cascades guarantee a strong and provable anonymity level for $l = O(\log k)$ [13], where the constant hidden by big 'Oh' does not depend on N. A similar result holds for mix cascades with RPC, but with a bigger constant.

6 Implementations issues

The Chaum's visual voting scheme requires special printers, which are able to print on transparent plastic cards, on both sides simultaneously. The system presented by Van de Graaf uses regular printers and paper receipts, but requires existence of an additional infrastructure (authorities that create and verify encoded votes). Our solution uses also regular paper printers, but no additional authorities are required – the codes are produced directly by the voting machines.

In the Brazilian system of Van de Graaf, there is also a problem of delivering prepared votes into the voting machines. Another problem is that the voting machines are connected to the network. This might be risky, since hacking voting machines could be the target of many attacks. In our solution voting machines are off-line, so attacks are possible only during tallying process. However, a successful attack on a tallying authority does not help to manipulate the election result.

"Onions" on the voting cards can be encoded in many ways. Depending on needs or technical possibilities one can use: bar codes (then bar code readers are needed), numerical representation (then OCR program for scanning is needed) or graphical representation, where the color of a pixel corresponds to the bit of the onion.

Our scheme can be slightly modified and then used for implementing elections via Internet. For this purpose some elements of previously described ballots have to be replaced by their cryptographic commitments. When the original protocol uses overprinting of a particular data, the Internet system does not reveal the contents related to a commitment of this data. If according to our protocol, some data remains uncovered (for example, the onions to be checked), then the Internet system reveals its value. Such an approach is particularly attractive for scenarios in which some voters use voting machines, while the other voters choose to vote online. Essentially the same procedure in both cases contributes well to the overall system quality.

7 Conclusions

The electronic voting scheme presented in this paper has the same advantages as Chaum's system. On the other hand, we have reduced demands on special hardware while the voters trust should be preserved even if no visual cryptography is used - the voter can trust the printed data as well. It is even less mysterious and therefore may be more acceptable by an average voter.

We also reduce the size of the messages processed if a high level of anonymity is to be achieved. Despite that in our scheme a voter sends four onions for a single vote, we can significantly reduce the number of tallying authorities to such a number l that with an overwhelming probability at least one out of l authorities is honest. For the Chaum's scheme, the number of tallying authorities is $O(1)$, however the size of this constant might be quite large from the practical point of view.

A still unsolved problem is the case of elections where the list of candidates is very long (for some voting systems each party presents a list of candidates, a voter chooses only one name from only one list). Then simply there is no enough place for all onions concerned on a single sheet of paper.

References

1. Berman, R., Fiat, A., Ta-Shma, A.: Provable Unlinkability Against Traffic Analysis. Financial Cryptography'2004, LNCS 3110, Springer-Verlag, 266-280.
2. Chaum, D.: Secret-Ballot Receipts and Transparent Integrity. Better and less-costly electronic voting and polling places.
3. Chaum, D.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM 24(2), 84-88, 1981.
4. Fairbrother, P.: An Improved Construction for Universal Re-encryption. Privacy Enhancing Technologies'2004, LNCS , Springer-Verlag.
5. Frankling, M., Haber, S.: Joint Encryption and Message-Efficient Secure Computation. Journal of Cryptology 9.4, 217-232, 1996.
6. Goldschlag, D.M., Reed, M.G., Syverson, P.F.: Hiding Routing Information. Information Hiding '1996, LNCS 1174, Springer-Verlag, 137-150.
7. Gomuł kiewicz, M., Klonowski, M., Kutył owski, M.: Rapid Mixing and Security of Chaum's Visual Electronic Voting. Computer Security- ESORICS 2003, LNCS 2808, Springer-Verlag, 132-145.
8. Golle, P.: Reputable Mix Networks. To appear in proceedings of Privacy Enhancing Technologies (PET) 2004, LNCS , Springer-Verlag, Springer-Verlag.
9. Golle, P., Jakobsson, M., Juels, A., Syverson, P.: Universal Re-encryption for Mixnets. CT-RSA'2004, 163-178.
10. Golle, P. Jules, A.: Parallel mixing. ACM Conference on Communications Security. ACM Press, 2004 (*http://crypto.stanford.edu/ pgolle/papers/parallel.pdf*)
11. Jakobsson, M., Juels, A., Rivest, R.L.: Making Mix Nets Robust For Electronic Voting By Randomized Partial Checking. USENIX Security Symposium 2002, 339-353.
12. Gomuł kiewicz, M., Klonowski, M., Kutył owski, M.: Onion Routing Based On Universal Re–Encryption Immune Against Repetitive Attack. To appear in proceedings of Workshop on Information Security Applications (WISA)'2004, LNCS , Springer-Verlag.
13. Klonowski, M., Kutył owski, M.: Provable Anonymity for Network of Mixes. Manuscript.

14. Klonowski, M., Kutyłowski, M., Lauks, A., Zagórski, F.: Universal Re-Encryption of Signatures and Controlling Anonymous Information Flow. To appear in proceedings of WARTACRYPT'2004, Tatra Mountains Mathematical Publications.
15. McGaley, M.: Report on DIMACS Workshop on Electronic Voting - Theory and Practice. (*http://dimacs.rutgers.edu/SpecialYears/2003_CSIP/reports.html*)
16. Neff, C.A.: A Verifiable Secret Shuffle and its Application to E-Voting ACM Conference on Computer and Communications Security 2001: 116-125.
17. Schnorr, C.P.: Efficient Signature Generation by Smart Cards. Journal of Cryptology 4, 161-174, 1991.
18. Van de Graaf, J.: Adapting Chaum's Voter-Verifiable election scheme to the Brazilian system. (*http://www.ppgia.pucpr.br/ maziero/pesquisa/wseg/2004/*)